
Monser
sinclair
48K
Spectrum
N.º 10
AÑO I
REVISTA
MENSUAL
COLECCIONABLE
P.V.P.
395
IVA inc.
AMERICAN
Billiard
Análisis Soft:
– TASWORD TWO
– POPEYE CON MAPA
Curso Código Máquina
Cap. X
RUTINA DE RENUMERACION
DE PROGRAMAS BASIC
Móvoa



MONSER, S.A.
Calle Argos, 9
28037 MADRID
Telf.: 742 72 12/96

ORDENA TU ORDENADOR



Ahora Vd. puede tener todo su equipo de ordenador en un gabinete de estilo con tres elegantes niveles. No más desórdenes de cables ni de periféricos. Además su equipo estará más protegido.

NO PIERDA ESTA OPORTUNIDAD UNICA

Tendrá espacio a su alcance para hardware y software.

Dispondrá de una unidad de puente de 56,5 cm ancho, 17 cm de alto y 30,5 cm de fondo para su televisor o monitor.

Debajo de esta unidad hay espacio suficiente para guardar su ordenador, aparato de cassette o microdrive.

En una tercera unidad tiene amplio espacio para guardar cintas, diskettes, joysticks, revistas, libros, etc.

Se vende desarmado en una caja plana, es muy fácil de armar, utilizando solamente una llave ALLEN.

El gabinete se presenta en dos colores, NOGAL y ROBLE y tiene dimensiones que se ajustan a las necesidades de espacio y altura que Vd. requiere.

ANCHO 85,5 cm. • ALTO 79,5 cm. • FONDO 60 cm.

Y ADEMAS LOS INTERFACES PARA SU JOYSTICK, IMPRESORA O MICRODRIVE

Interface DK'Tronics
Doble salida en la parte superior. La primera para joystick tipo Kempston y la segunda para software con teclas 6, 7, 8, 9 y 0 o redefinición de teclas.
Ref. 30001. P.V.P. 3.760 ptas.

Interface Centronic.
Para impresora y microdrive en paralelo. Ref. 30010.
P.V.P. 11.358 ptas.

Para pedidos simplemente rellene el cupón.

LOS JOYSTICKS DE GRAN RESPUESTA Y DURABILIDAD









Por favor, envíenme:(marco con una X):

	REF.	COLOR	CANTIDAD	PRECIO	SUBTOTAL
GABINETE	40005	Nogal		8.975	
	40005	Roble		8.975	
JOYSTICK	30007			1.900	
	30003			2.598	
	30008			3.724	
	30005			2.450	
INTERFACE	30001 30010			3.760 11.358	

Gastos de envío gabinete 800 ptas., joysticks 200 ptas. ....

TOTAL ....

Forma de pago:

☐ Talón bancario a nombre de MONSER, S.A.

☐ Giro postal núm. ....

☐ Contra reembolso

Nombre ....

Domicilio ....

Ciudad....................................... Provincia ....

C.P. .... Telf.: ....

FIRMA



Sumario:

AÑO I - N.° 10 - 1985 -

DIRECTOR
José Nieto Rubio

COORDINADOR
Félix Santamaría Avila

SUPERVISOR SOFTWARE
Gustavo Cano Muñoz

DISEÑO
Marco Antonio Díaz

REDACCION Y COLABORADORES

Victoria Aguilar
Javier González

PORTADA
Mauro Novoa

EDITA
MONSER, S. A.

DIRECTOR EDITORIAL
J. L. Cano Regidor

REDACCION, ADMINISTRACION Y PUBLICIDAD
Argos, 9
28037-MADRID
Tel. 742 72 12/96

PUBLICIDAD Y SUSCRIPTORES
Yolanda Bardillo

FOTOCOMPOSICION
CRISOL, S. A.
Virgen del Val, 48

FOTOMECANICA
IMAGEN

IMPRIME
GRAFICAS IBARRA

DISTRIBUCION
R. B. Z.

Depósito legal: M-10.328-1985

Se solicitará control O.J.D.

ANALISIS

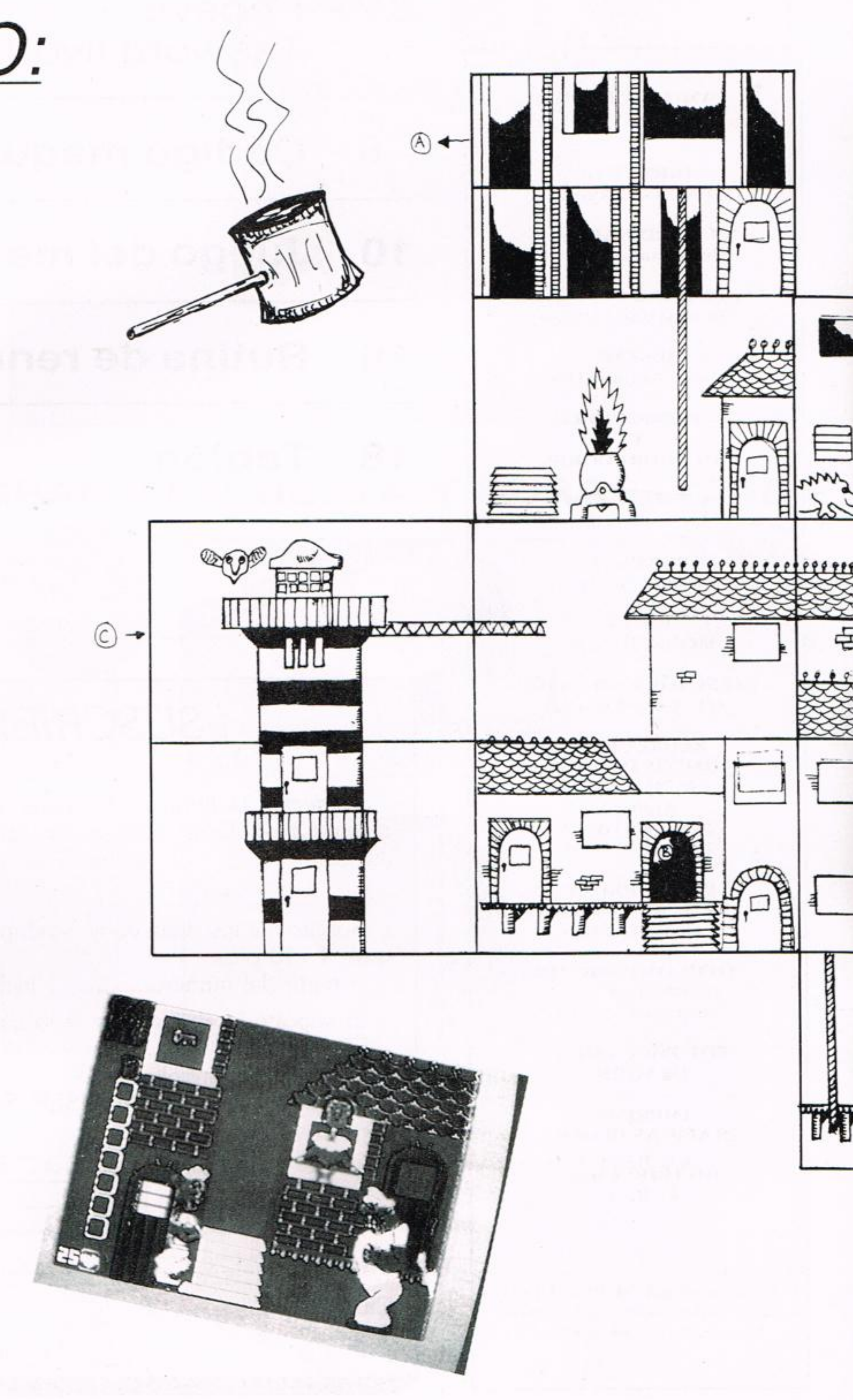
COMENTARIO: POPEYE

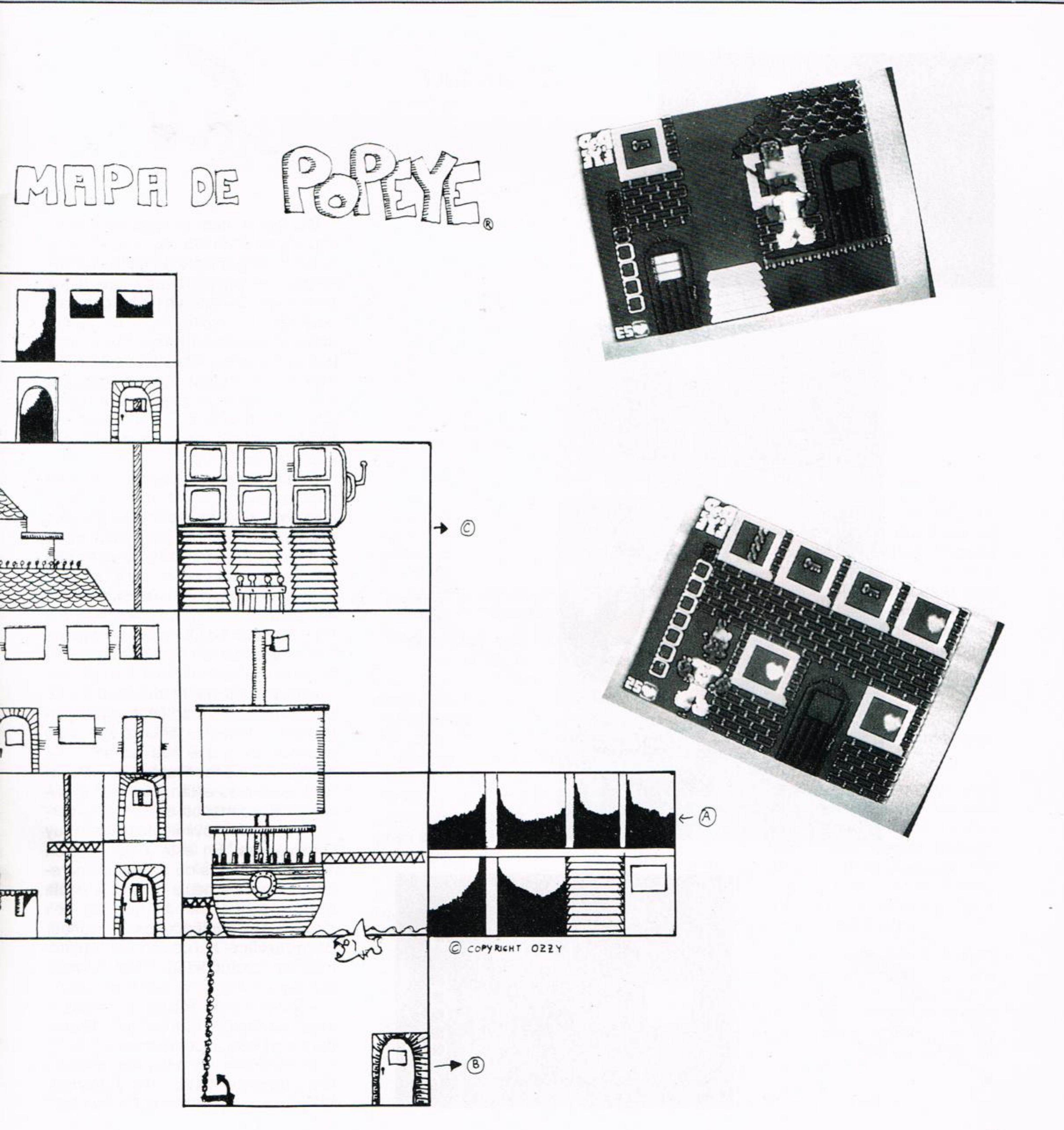
Nos encontramos frente a un programa en el cual los gráficos se han tratado con gran originalidad. El juego, como últimamente se ha puesto de moda, es de un famoso personaje de dibujos animados, «Ojo saltado», o para entendernos, Popeye.

El juego consiste en obtener 25 corazones distribuidos por los distintos escenarios y llevárselos a Olivia, la huesuda compañera de Popeye y sin los cuales perderemos su amor y por lo tanto, el juego. Asimismo, nos encontramos a algunos de los personajes del comic que nos harán la vida imposible, a saber: Brutus, la Bruja, el Buitre, etc. Hay que alejarse de ellos, ya que con un simple contacto harán que Popeye muerda el polvo.

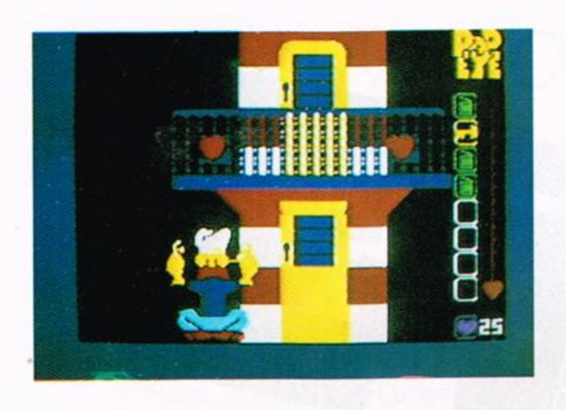
Pero hablaremos ahora más lentamente de la misión de Popeye. El juego se desarrolla en una ciudad portuaria (con faro y todo). En las habitaciones, tejados, sótanos, etc., encontraremos 3 elementos fundamentales: corazones, espinacas y llaves.

Los corazones, como ya hemos dicho, tienen que ser recogidos y llevados a la casa de Olivia (a la derecha del faro, primera ventana). Como observarás, hay corazones que no se pueden coger, ya que algunas puertas nos lo impiden. Pues bien, aquí entran en juego las llaves. Debes recolectar llaves con las que puedas abrir habitaciones y obtener corazones. Como ejemplo podemos decir que si cogemos la llave que se encuentra en la puerta de la cuerda, podremos abrir la puerta del faro, pudiendo así subir a la parte superior del faro, obtener otra llave y abrir la puerta del piso intermedio del faro para así obtener un corazón y un bote de espinacas.

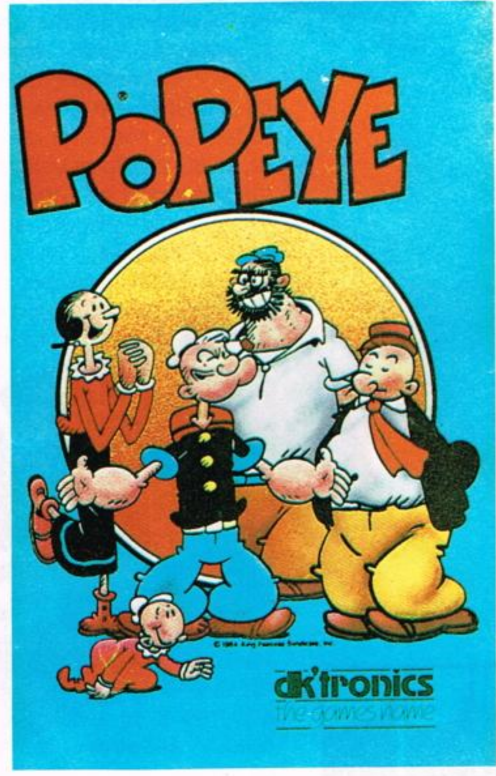



MAPA DE POPEYE®
→ C
← A
© COPYRIGHT OZZY
→ B

ANALISIS









Decíamos que el segundo objeto importante eran las espinacas. Estas te serán de gran utilidad cada vez que tengas un encontronazo con algún personaje. Cuando ya no tengas más reservas de espinacas y te «sacudan», se acabará el juego. Por lo tanto haz el mayor acopio posible de ellas. Aparte de estos objetos básicos, encontrarás otros que de una u otra forma te ayudarán a completar tu misión... serás tú el que deberás encontrar su función.

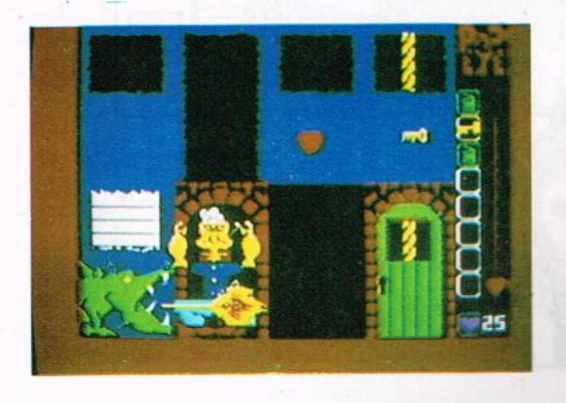
Debes tener en cuenta que sólo podrás llevar encima un total de 8 objetos, y podrás saber lo que tienes observando los 8 cuadrados que hay a la derecha de la pantalla. A la derecha de estos cuadrados está lo que podríamos denominar «amorometro», es decir, el aparato que nos indica el nivel de amor de Olivia hacia Popeye. Sabes que cuando esté bajo, deberás llevarle corazones a Olivia o perderás.

Hablaremos ahora del tratamiento de los gráficos. Lo primero que observamos es que las figuras son más grandes de lo que nos tenían acostumbrados los fabricantes de software. Los colores están muy bien elegidos y los escenarios, así como los personajes y sus movimientos están muy conseguidos. En este juego existen «capas» o zonas de pantalla. Podremos movernos hacia delante y hacia atrás (aunque parece que no nos movemos) pudiéndonos esconder detrás del faro, así como ir por laberintos con profundidad. Muy original, aunque difícil de entender al principio.

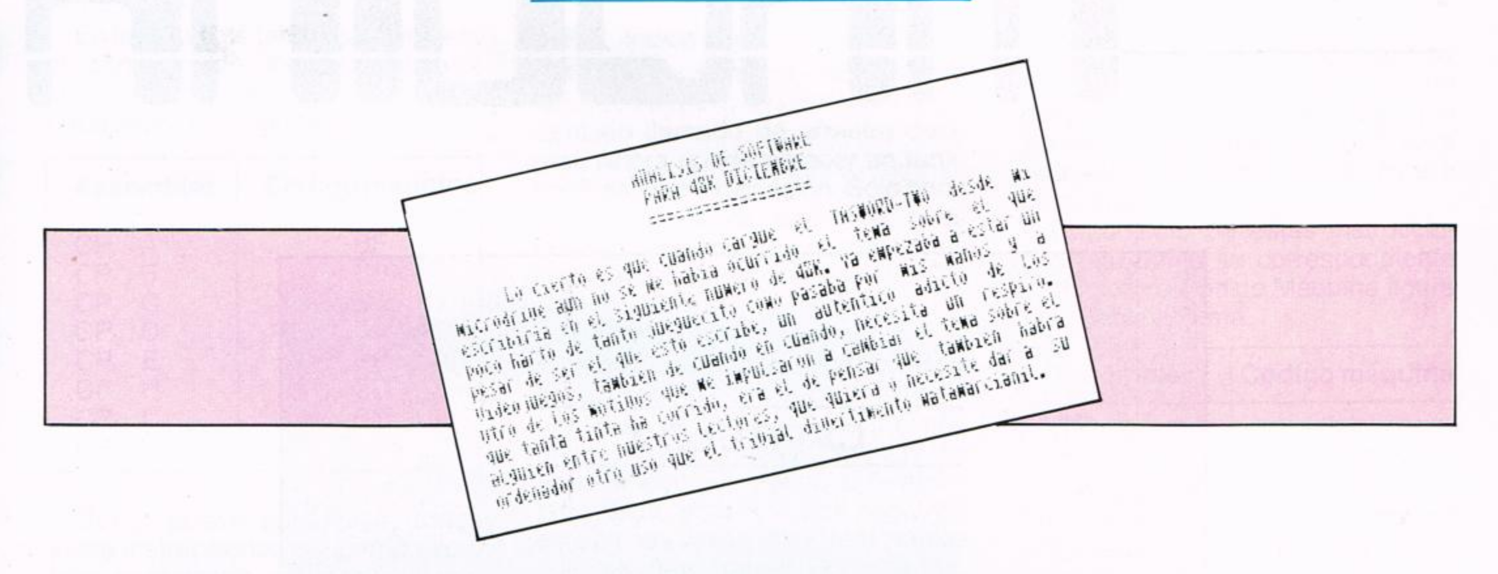
Adjunto a este artículo te damos el mapa de Popeye que sin duda te ayudará a conseguir corazones.

En definitiva, un juego muy original, con buenos gráficos y muy adictivo. Un gran acierto de la casa Dk'Tronics.





TASWORD-TWO


ANALISIS DE SOFTWARE
PARA 48K DICIEMBRE
==================

Lo cierto es que cuando cargue el TASWORD-TWO desde mi
microdrive aun no se me habia ocurrido el tema sobre el que
escribiria en el siguiente numero de 48K. Ya empezaba a estar un
poco harto de tanto jueguecito como pasaba por mis manos y a
pesar de ser el que esto escribe, un autentico adicto de los
videojuegos, tambien de cuando en cuando, necesita un respiro.
Otro de los motivos que me impulsaron a cambiar el tema sobre el
que tanta tinta ha corrido, era el de pensar que tambien habra
alguien entre nuestros lectores, que quiera o necesite dar a su
ordenador otro uso que el trivial divertimento matamarcianil.

Lo cierto es que cuando cargué el TASWORD-TWO desde mi microdive aún no se me había ocurrido el tema sobre el que escribiría en el siguiente numero de 48K. Ya empezaba a estar un poco harto de tanto jueguecito como pasaba por mis manos y a pesar de ser el que esto escribe, un auténtico adicto de los videojuegos, también de cuando en cuando, necesita un respiro. Otro de los motivos que me impulsaron a cambiar el tema sobre el que tanta tinta ha corrido, era el de pensar que también habrá alguien entre nuestros lectores, que quiera o necesite dar a su ordenador otro uso que el trivial divertimento matamarcianil.

Dado el precio al que algunos de nosotros adquirimos en su día el maravilloso SPECTRUM, no sería rentable, si por nuestro trabajo necesitásemos una máquina de escribir, tener que comprarla, y muchísimo menos si tenemos en cuenta que la máquina de escribir más barata que yo conozco, cuesta alrededor de las 14.000 pesetas. Ese es uno de los motivos, quizá el primero, al empezar a trabajar con un procesador de textos. La verdad sea dicha, ese sólido argumento se nos viene abajo desde el primer momento en el que empezamos a profundizar en las posibilidades que nos ofrece cualquier procesador de textos y en particular el TASWORD-TOW.

Reconozco que es una casualidad, pero se da el caso de que mi mujer es por profesión, secretaria y para más detalles, de dirección, lo que nos da idea de la presentabilidad que deben tener sus escritos. Pues bien, como decía, fue por casualidad que un día, mientras le contaba las excelencias de este procesador, cosa que hago, con gran aburrimiento por su parte, con todo programa que llega a mis manos, noté cómo ella comenzaba a interesarse por el TASWORD-TWO, hasta el punto de rogarme encarecidamente que lo cargase un momento en la máquina para que ella pudiera dar crédito a sus oídos.

Tras algunos minutos de prueba, se volvió hacia mí, y me dijo: «Chapeau..., por fin descubro que tu maquinita roba-horas, sirve para algo más que para hacerme pasar las tardes en casa».

Ya comenté antes que ella es secretaria, lo que no comenté es que en su puesto de trabajo dispone de una máquina sólo y exclusivamente dedicada a «procesar textos», y aunque la máquina en cuestión seguro que sabe hacer muchas más cosas, nadie se las pide, pues el Software para cualquier ordenador profesional es excesivamente caro.

«¡¡Pero si es exactamente igual que el mio...!!», exclamó.

Y es cierto, el TASWORD-TWW es un procesador perfectamente llamado profesional. Con él podemos escribir, centrar, mover, copiar, buscar, tabular, salvar, cargar, margear, ir al principio, ir al final; borrar líneas, palabras, letras, y, lo más importante, imprimir los resultados.

Y para más prestaciones, que en un procesador muchísimo más caro tiene que hacerlo un experto, aquí lo haces tú mismo sin necesitar ningún conocimiento de informática, estoy hablando del formateo del buffer de la impresora.

Esto es formidable.

CODIGO MAQUINA

CAPITULO X

1.—INTRODUCCION

En este capítulo vamos a contemplar las instrucciones de comparación y búsqueda dentro del juego de instrucciones del Z-80. Incluiremos también una instrucción que en cierto modo incluye una comparación y que se utiliza frecuentemente para formar bucles.

Si quisiéramos realizar una analogía entre las instrucciones que trataremos en este capítulo y las del lenguaje Basic que pudiéramos hacerlas corresponder, diríamos que en este capítulo contemplaríamos los bucles FOR... NEXT y los IF, principalmente. Sin embargo, para explicar las instrucciones de búsqueda que vamos a ver, en Basic tendríamos que recurrir a crearnos rutinas compuestas por bucles y comparaciones. No obstante hemos de señalar que las búsquedas mediante una instrucción en lenguaje Assembler están limitadas a encontrar un sólo carácter.

2.—Instrucciones de comparación

El formato de estas instrucciones en lenguaje Assembler es:

CP r

en donde,

r representa a alguno de los registros simples, A, B, C, D, H, y L, o a una constante cuyo valor está comprendido entre 0 y 225 en decimal (# OO - # FF).

En la siguiente tabla podemos ver el conjunto de instrucciones de comparación y su correspondiente traducción a Código Máquina.

Assembler	Código máquina
CP A	BF
CP B	B8
CP C	B9
CP D	BA
CP E	BB
CP H	BC
CP L	BD
CP n	FE xx

Como puede apreciarse, todas estas instrucciones ocupan un solo byte de memoria, excepto la última, que ocupa dos, por tener que llevar el valor a comparar de un modo directo.

La función de estas instrucciones es la de realizar una comparación entre el operando que aparece en la instrucción (registro simple o constante) y el contenido del registro Acumulador.

La forma de llevar a cabo esta comparación es por medio de una resta. Pero no hay motivos para asustarse porque el contenido de todos los operandos, tanto el Acumulador como el registro simple, no se modifican. La sustracción, que es del tipo A-r, se realiza en la Unidad Lógica Aritmética del microprocesador y lo único que sufre modificación, cara al programa, son los indicadores o flags.

En efecto, según cual sea el resultado de esta resta, los indicadores o flags se modificarán de la siguiente manera:

C *Indica que:*

El carry flag se activa si el número a comparar, contenido en el registro simple, o la constante, es *mayor* que el contenido del Acumulador. Al hacerse la resta se produciría un desbordamiento (overflow) y el bit C se activaría.

Si C =1 es que r > A
Si C= 0 es que r ≤ A

Z *Indica que:*

El bit de cero se activa si el resultado de la resta es cero. Esto indicaría indudablemente que ambos operandos son *iguales*.

Si Z = 1 r = A
Si Z = Ø r < >

P/V *Indica que:*

Este flag se pone «1» si no existe acarreo negativo desde el cuarto bit, también llamado de arrastre decimal. Ahora puede parecer un tanto confusa esta explicación. Sólo pedimos un poco de paciencia pues cuando veamos las instrucciones decimales notaremos su interés.

S *Indica que:*

Este bit de signo indica por su estado, activado o no, si el contenido del registro simple, o constante, son menores, o no, que el contenido del Acumulador. En efecto, si el resultado de la resta A − r es negativo, este bit se *activa* y por lo tanto indica que *r* es *mayor* que *A*. Por el contrario, si *A* es igual o mayor que *r*, el bit se desactiva.

N *Indica que:*

El indicador de suma/resta se pone activado siempre que se ejecuta esta operación, porque de hecho se trata de una resta.

Más de uno se habrá preguntado el motivo por el cual existe una instrucción tan tonta como la de comparar al registro Acumulador consigo mismo (CP A). Si volvemos a revisar el estado en que quedan los flags, tal vez podamos deducir que, lo único que hace esta instrucción, es dejar todos los indicadores, menos el N, a cero. Este es el uso que se le da normalmente.

Otro grupo de instrucciones de comparación es el constituido por aquellas en las que el byte a comparar con el contenido del registro Acumulador viene determinado, no ya por el contenido de un registro o una constante, sino por una dirección que viene reflejada por un registro par.

La función de dichas instrucciones es la misma que las de las explicadas anteriormente, y los flags se activan y desactivan siguiendo las mismas reglas. Unicamente varía en la forma de obtener el byte a comparar, que en lugar de ser de un modo inmediato en este grupo se consideran el direccionamiento indirecto (por el contenido de un registro par) e indexado por el contenido de los requisitos IX e IY más un desplazamiento.

El formato de este grupo en el lenguaje Assembler es:

CP (p)

en donde,

p representa al par de registros HL, o a los registros IX ó IY con un desplazamiento *d* que hay que sumar al contenido IX ó IY para obtener la dirección real del byte a comparar.

El conjunto de estas instrucciones, así como su correspondiente traducción a Código Máquina figura en la tabla adjunta.

Assembler	Código máquina
CP (HL)	BE
CP (Ix + d)	DD BE xx
CP (Iy + d)	FD BE xx

En este caso, el valor de xx es el valor del desplazamiento, y como ya habíamos visto para el caso de los saltos relativos, puede tener un valor comprendido entre −128 y +127.

La longitud de cada instrucción puede deducirse de la tabla sin necesidad de que ahondemos en más detalles.

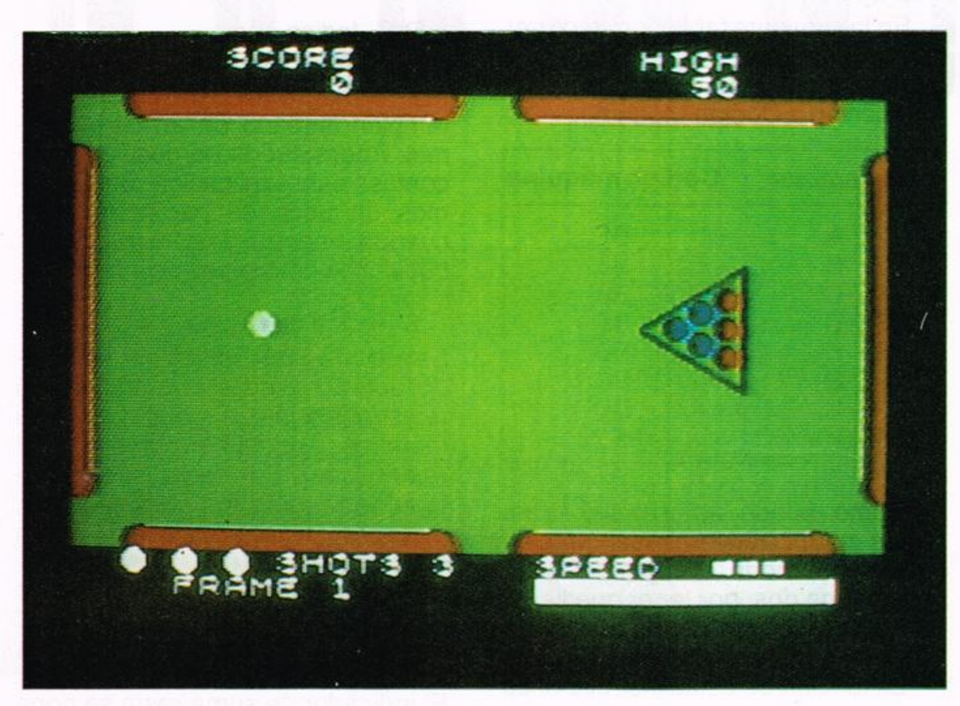
Lo normal es que tras una pregunta, y obtener su correspondiente respuesta, tomemos una decisión en función de dicha respuesta. La opción de seguir uno u otro camino, la podemos adoptar mediante la utilización de las instrucciones de bifurcación condicional, tanto absolutas como relativas, que ya hemos visto en capítulos anteriores.

Como ejercicio sencillo y fácil de llevar a la práctica, aconsejaríamos que se volviera a estudiar la rutina TEXTO que ya hemos visto en el anterior capítulo. Podíamos ver la posibilidad de hacer algunas sustituciones en consonancia con la función que hemos visto que realiza esta instrucción. Por si puede considerarse como una pista, aconsejamos que se repasen los motivos por los cuales se realizaba una resta y posteriormente una suma.

JUEGOS DEL MES

American Billiards

Nadie esperaba que reaccionásemos tan rápido y lo cierto es que nosotros tampoco. Apenas pasaron diez minutos desde que recibimos el aviso, cuando ya caminábamos lenta, muy lentamente, por la calle, armados tan sólo con nuestros palos, abarcando de una a otra acera. No necesitábamos pedirle a nadie que se hiciera a un lado,... ni los gatos resistían vernos de cerca y en cuanto olían nuestra proximidad se les erizaban los pelos del lomo como si de un jarro de agua fría se tratase nuestra presencia. Este lado de la ciudad era muy peligroso y más especialmente a esta hora de transición entre el día y la noche, esa hora donde todo pierde el color y el volumen para convertirse en frías siluetas que no en todos los casos se distinguen con claridad; en





momentos como este, cualquier cosa puede suceder, por eso lo más juicioso no es marchar nunca en solitario. La soledad es el peor de los males, es aburrida, sórdida, pesada y en estos casos incluso peligrosa.

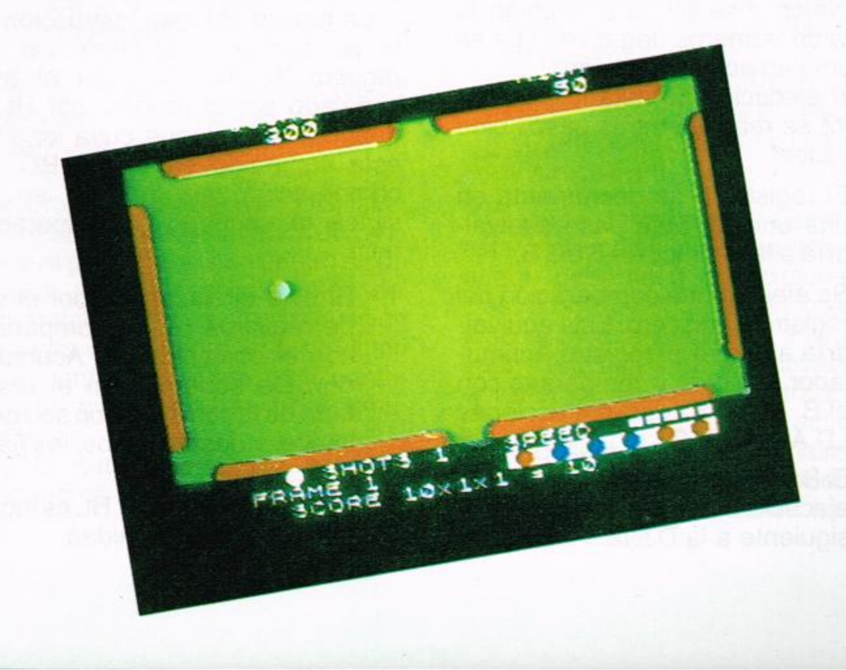
A medida que avanzábamos, se oían a ras de suelo unos leves grititos y unos ojos brillantes, extraños, centelleaban en la oscuridad horizontal de la abertura de las alcantarillas; nos miraban como queriendo transmitir algo, como comunicándose solidarias con nosotros, como sintiéndonos de alguna manera hijos suyos, o al menos ahijados, pues seguramente más de una, o quizá alguna de sus abuelas, asistieron a nuestro bautizo. Una patada sobre el pavimento fue suficiente para que en una alborotada orgía de gritos, desaparecieran los cientos de pequeños ojos que ya comenzaban a rodearnos.

La calle gris desentonaba con los «graffitis» de las paredes. El arte popular, junto con el de algún entendido en bellas artes y el de algún que otro maldito obseso, convertían cada plano vertical en una auténtica verbena en la que podrías pasar perfectamente varias horas descubriendo cosas nuevas y por lo tanto sin aburrirte ni un ápice.

Tras la última esquina nos encontrábamos en el centro de un tenebroso callejón y al fondo del mismo unas luces de neón nos anunciaban que estábamos muy próximos al garito de donde partió la amenaza. Ya hacía bastante tiempo que nos sentíamos presos de ese desagradable hormigueo que se te pone en la nuca cuando alguien te observa sin apartar la mirada; era lógico, pues hacía más de media hora que habíamos entrado en su territorio. Cada paso que dábamos nos acercaba un poco más al epicentro del terremoto que había sido capaz de movilizar a diez de los mejores jugadores de billar de la ciudad.

Desde la puerta del oscuro antro, apenas se podían ver signos de vida en el interior del mismo, tan sólo bajo las mortecinas luces que había sobre cada mesa de billar se apreciaba el movimiento lento y pesado de cualquiera de las bolas de sebo que se pasaban una hora tras otra con el tabaco en una mano y las tizas en la otra, tanto afilando la punta como puliendo la superficie.

—¿Ya estás aquí? —te dijo con voz ronca uno de esos amasijos de carne—, eso significa que aceptas el reto, ¿no es así? —decía sin dejar de echar humo por esa bocaza...—, el que gane podrá considerarse el rey de la ciudad. Sólo de ti depende colarlas todas, o dejar alguna para tu contrario.



3.—Decremento y y salto DJNZ

Esta instrucción es una micro-rutina, lo mismo que algunas otras del microprocesador Z-80. Esto quiere decir que no realiza una sola función elemental, sino que realiza lo que un conjunto de instrucciones elementales. En efecto, esta que nos ocupa realiza un decrecimiento del contenido del registro B y si el resultado no es igual a cero realiza un salto relativo. Como puede apreciarse es una reunión de instrucciones conocidas, que podrían haberse realizado mediante una rutina en Assembler, compuesta por unas pocas órdenes. No osbtante, el Z-80 nos proporciona esta rutina, que se utiliza con muchísima frecuencia, en una única instrucción con la ventaja de no ocupar más que dos bytes y además realizarse con mayor rapidez que por medio de varias instrucciones, pues la Unidad de decodificación y Control del microprocesador no interviene más que una sola vez.

El formato de esta instrucción en Assembler, así como su traducción a Código Máquina es:

Assembler	Código máquina
DJNZ d	10 xx

en donde,

d, y su equivalente, xx, son cifras comprendidas entre +127 y −128, rigiéndose por las mismas normas y que ya comentamos al hablar de las instrucciones de saltos relativos. Para la traducción a hexadecimal del valor *d* se utilizará también la tabla de números negativos que se adjuntó en aquella ocasión.

La ejecución de esta instrucción DJNZ se realiza en tres pasos bien definidos:

a) El registro B se decrementa en una unidad. Este paso equivaldría a la instrucción DEC B.

b) Se efectúa una comparación del registro B con cero. Esto equivaldría a cargar el registro Acumulador con cero y compararlo con el B. Es decir, dos instrucciones LD A,Ø y CP B .

c) Si B es igual a cero se continúa la ejecución del programa en la siguiente a la DJNZ. Por el contrario, si B fuera *distinto* de cero se realiza un salto relativo a la instrucción señalada por el desplazamiento *d.* Esta fase correspondería a una instrucción JR NZ, d.

Aun a costa de parecer pesados por insistir tanto, no queremos dejar pasar la ocasión de señalar dos cosas:

1.° Si cuando el programa llega a ejecutar esta instrucción el registro B tiene un cero, la bifurcación, o salto al desplazamiento se produce, porque si a cero le restamos una unidad el resultado es −1, diferente de cero.

2.° El valor de *d* es un desplazamiento relativo. Es decir, un número que hay que *sumar* o *restar* al valor que en ese momento tenga el Pc (program counter) y que en el momento de hacer la suma, el Pc apunta precisamente a la instrucción siguiente a la DJNZ.
En otras palabras, la instrucción DJNZ Ø no realiza salto alguno sea cual sea el resultado de la comparación de B con cero.

4.—Comparar e incrementar CPI

El formato de esta instrucción en Assembler es

CPI

y su traducción a Código Máquina es # ED A1

La función de esta instrucción es la de comparar el contenido del registro Acumulador con el byte señalado por el registro par HL, y dentro de un bloque cuya longitud está contenida en el par BC. La comparación se realiza en la siguiente secuencia de operaciones:

1.° El byte direccionado por el par de registros HL es comparado con el contenido del Acumulador. De acuerdo con el resultado de la comparación son activados, y desactivados, los flags Z, S, y H.

2.° El contenido del par HL es incrementado en una unidad.

2.° El contenido del par BC es decrementado en una unidad. En este momento, si el contenido del Acumulador y el byte direccionado por el par HL son iguales el indicador Z se activa, y en caso contrario, se desactiva. El indicador de P/V se pone a cero si el par BC es igual a cero y se pone a 1 en caso contrario.

Durante la ejecución de esta instrucción el flag Carry no se altera.

Esta instrucción, que también pertenece al grupo de las micro-rutinas puede utilizarse para buscar un byte determinado, o para ir analizando, byte a byte, un bloque comparándolo con un valor determinado.

Debemos llamar la atención sobre un pequeño detalle: tras la ejecución de la rutina, el par HL apunta al carácter siguiente a comparar y no sobre el recién comparado.

5.—Comparar y decrementar CPD

Complementaria de la instrucción que hemos visto en el párrafo anterior, el Z-80 nos proporciona otra instrucción micro-rutina que permite realizar la comparación del contenido del requisito Acumulador con el byte direccionado por el contenido del registro par HL, y todo ello dentro de un bloque cuya longitud está contenida en el registro par BC. La única diferencia con respecto a la instrucción que vimos en el anterior párrafo, la CPI, es que en lugar de incrementar el contenido del par HL, lo que esta instrucción hace es decrementarlo, o sea, restarle una unidad.

El formato en Assembler de esta instrucción es

CPD

y su traducción a Código Máquina es # EDA9.

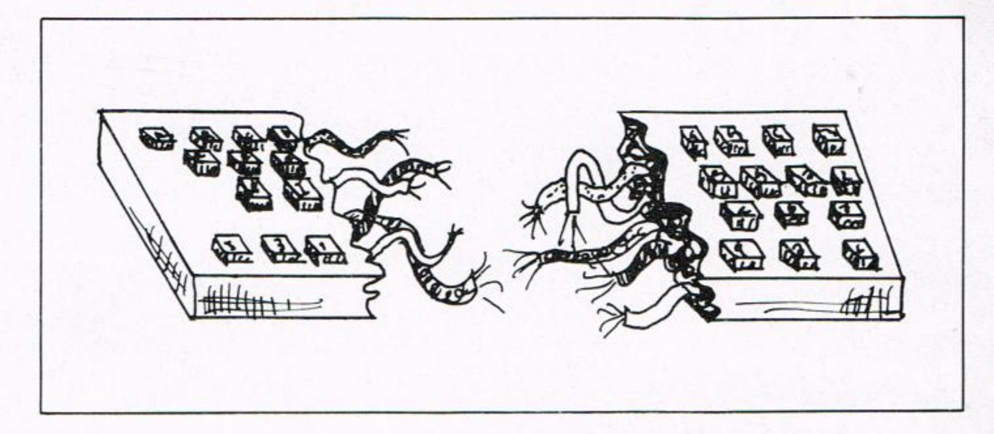
La ejecución de todas las operaciones que conlleva la instrucción se realiza en la siguiente secuencia:

1.° El byte situado en la dirección indicada por el contenido del par HL es comparado con el contenido del registro Acumulador. Los flags Z, S, y H son coloca-

dos de acuerdo con el resultado de la operación.

2.° El contenido del registro par HL es decrementado en una unidad.

3.° El contenido del par BC es también decrementado en una unidad. Si el contenido de este registro es igual a cero el indicador P/V se pone a cero, pero si no fuera así, este indicador saldrá de esta instrucción con un uno.



4.° Si el par BC es igual a cero, el indicador P/V es puesto a uno.

5.° Si la comparación realizada en el primer paso no ha detectado igualdad, y el contenido del registro par BC no es igual a cero, se volverá a la ejecución del paso 1.°.
Si la comparación refleja igualdad, o el registro par BC es cero, el control del programa pasa a la instrucción siguiente a la CPIR.

Habrá que tener en cuenta que, tras la comparación, el par HL ha sido decrementado, por lo que si se ha producido la igualdad ha sido con el byte posterior al señalado por HL.

6.—Comparar incrementar y repetir CPIR

Para la búsqueda de un valor dentro de un bloque, el juego de instrucciones del Z-80 dispone también de una instrucción concreta, perteneciente al grupo de las que llamamos micro-rutinas, pues realiza una serie de funciones en una sola instrucción.

El formato Assembler de esta instrucción es

CPIR

y su traducción a Código Máquina es ≠ EDB1.

Como ya hemos indicado, esta instrucción realiza la búsqueda de un byte cuyo contenido sea igual al del contenido del registro Acumulador, dentro de un bloque de datos cuyo comienzo está contenido en el par HL, en el momento de iniciar la ejecución de la instrucción, y cuya longitud está contenida en el registro par BC.

La secuencia de operaciones que se realizan al ejecutar esta instrucción es la siguiente:

1.° El byte direccionado por el contenido del registro par HL es comparado con el contenido del Acumulador. Los indicadores Z, S y H son actualizados de acuerdo con el resultado de la operación.

2.° El registro HL es incrementado en una unidad.

3.° El registro BC es decrementado en una unidad.

Cuando termina de ejecutarse esta instrucción ya sabemos que ha sucedido una de estas dos condiciones: o no hay un carácter igual al contenido en todo el bloque (indicador P/V activado, y el flag Z desactivado), o en efecto ha encontrado un carácter igual (flag Z activado, P/V desactivado). Con tal de analizar el contenido del flag Z, mediante una instrucción de salto condicional, podremos diferenciar cuál de las dos condiciones se ha cumplido.

7.—Comparar, decrementar y repetir

También para la instrucción de Comparar, Incrementar y Repetir existe una complementaria en la que en lugar de incrementar se decrementa. Podríamos decir que, mientras la anterior instrucción buscaba el primer byte que dentro de un bloque era igual al contenido del Acumulador, con esta instrucción que nos ocupa se busca el último byte, dentro del bloque, cuyo contenido es igual al del registro Acumulador.

El formato Assembler de esta instrucción es

CPDR

y su traducción a Código Máquina es ≠ EDB9.

De un modo «académico» podíamos decir que la función de esta instrucción es la de buscar un byte cuyo contenido sea igual al del registro Acumulador, dentro de un bloque cuyo final está direccionado por el registro par HL y cuya longitud está contenida en el registro par BC.

La secuencia de operaciones que se realizan al ejecutar esta instrucción es la siguiente:

1.° El byte direccionado por el contenido del registro par HL es comparado con el contenido del Acumulador. Los indicadores Z, S y H son actualizados de acuerdo con el resultado de la operación.

2.° El registro par HL es decrementado en una unidad.

3.° El registro BC es decrementado en una unidad.

4.° Si el par BC es igual a cero el indicador P/V es puesto a uno.

5.° Si la comparación realizada en el primer paso no ha detectado igualdad, y el contenido del registro par BC no es igual a cero, se volverá a la ejecución del paso 1.°.
Si la comparación refleja igualdad, o el registro par BC es cero, el control del programa pasa a la instrucción siguiente a la CPDR.

Cuando termina de ejecutarse esta instrucción ya sabemos que puede deberse a dos circunstancias: o no hay un carácter igual en todo el bloque (indicador P/V activado y el indicador Z desactivado), o en efecto se ha encontrado un carácter igual (indicador Z activado y P/V desactivado). Para analizar el resultado bastará decidir, mediante una bifurcación condicional, a partir del indicador Z.

MONSER cada dia +

6 SuperCasettes
Full Memory
por solo 1.795 pts.

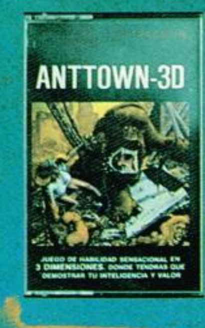




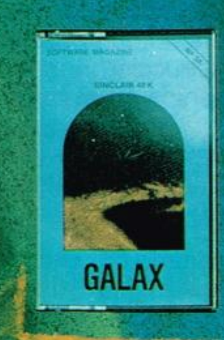
Libreto que contiene manual
de instrucciones en Castellano
+Aplicaciones para tu Spectrum

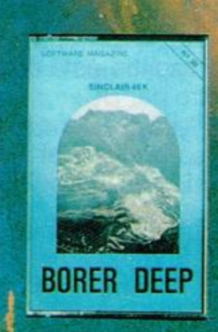




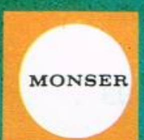








Ya a la venta en Kioscos,
Tiendas Especializadas y
Departamento de
Informatica del El Corte Inglés



MONSER S. A.
C/ Argos, 9 - 28037 Madrid .
Teléfonos: (91) 742 72 12 - 742 72 96

TYPE AND RUN

LA PRIMERA REVISTA CON CASSETTE PARA TECLEAR Y GRABAR

AÑO I – N.º 7

LISTADOS PARA

SPECTRUM

Rutina de renumeración
Matemáticas comerciales
Lunar ...

Navegar

COMMODORE

...tallas
Super boxeo

AMSTRAD

Policía de carretera

MENSUAL

Pídala en su kiosko o en tiendas especializadas

SUSCRIPCIONES: MONSER, S.A. C/ Argos, 9 - 28037 MADRID



Monser

RUTINA DE RENUMERACIO

Un buen programador no es aquel al que los programas le funcionan a la primera, o al menos no el mejor de todos; para mi forma de ver, un buen programador es aquel al que aunque un programa le funcione, no para hasta que contenga todos los detalles y mejoras que se le vayan ocurriendo a lo largo de varios días y... pasa lo que pasa, que llega un momento en que ya no es posible insertar ni una sola línea de programa más, pues ya hemos cubierto absolutamente todos los números que habíamos previsto en nuestro intervalo inicial; ha llegado el momento de reorganizar nuestro programa, pero no vamos a recorrernos instrucción por instrucción cambiándolas de número, sería absolutamente absurdo sobre todo si tenemos en cuenta que lo que hay delante de nuestras narices es un «ordenador», (sería de tontos ordenar a mano).

La rutina que a continuación se explica nos ayudará en esta empresa, agotadora si no dispusiéramos de ella.

En primer lugar teclearemos el listado número uno, que como su nombre indica, es el cargador de rutinas. A continuación daremos RUN y el programa nos pedirá la dirección donde queremos colocar nuestra rutina, que para más señas es totalmente reubicable, es decir, que la podremos poner donde queramos siempre y cuando no pisemos el área de Basic ocupada por nuestro programa; tras esta primera pregunta, el ordenador nos pedirá datos, que no son otros que los que aparecen en el listado número dos y que se introducirán en sentido horizontal, línea a línea y no columna a columna como podría parecer. Cuando ya hayamos terminado de introducir los valores del listado número dos, daremos un valor cualquiera superior a 255 con lo que nos saldremos del programa cargador. En este momento os recomendamos que salvéis los códigos introducidos con el comando:

SAVE «rutinanum» CODE XXXXX, 69Ø siendo XXXXX la dirección donde decidimos al principio del proceso colocar nuestra rutina.

Ahora teclearemos el programa que aparece con el nombre de Listado tres, y cargaremos con MERGE el programa de renumerar, y con un RUN 999Ø ejecutaremos el programa tres y contestaremos a todas sus preguntas, lo que nos llevará a la renumeración automática de nuestro programa Basic.

Esta rutina renumera no sólo las instrucciones, sino también todos los GOTO's, GOSUB's, RESTORE's, etc... salvo los que son operativos, que nos los informará la propia rutina por pantalla, con el número de instrucción y lugar ocupado en la misma. Un operativo es el obtenido por medio de la aplicación de fórmulas algorrítmicas, como por ejemplo GOSUB 1ØØ*N.

Listado 1

```
   1 REM cargador de rutinas
   2 INPUT "donde empezamos? ";d
es
   3 INPUT "introduce valor  ";b
yte
   4 IF byte>255 THEN STOP
   5 POKE des,byte
   6 LET des=des+1
   7 GO TO 3
```

Listado 2

205	107	13	42	123
92	229	42	125	92
229	205	124	0	59
59	225	229	1	16
1	9	94	35	86
237	83	123	92	35
94	35	86	237	83
125	92	24	42	124
2	109	0	68	2
14	1	84	2	33
1	68	2	167	1
84	2	86	1	196
1	192	1	223	0
82	2	199	0	89
2	84	2	234	0
145	2	239	0	255
255	225	229	1	25
0	9	94	35	86
35	122	254	255	40
19	235	193	197	9
235	213	94	35	86
35	235	9	193	113
35	112	235	24	226
225	42	83	92	126
230	192	194	218	197
1	255	255	126	230

N DE PROGRAMAS BASIC

192	32	10	35	35
94	35	86	35	3
25	24	241	205	43
45	237	75	123	92
205	43	45	205	233
52	56	27	239	4
56	237	75	125	92
205	43	45	205	233
52	56	12	239	15
56	205	162	45	33
240	216	9	48	12
205	197	22	33	1
0	34	123	92	34
125	92	167	42	125
92	237	91	123	92
237	82	34	69	92
42	83	92	35	35
35	34	176	92	34
93	92	42	123	92
237	91	69	92	25
34	69	92	175	50
71	92	231	253	52
13	254	58	40	248
254	13	202	178	197
254	234	202	239	197
254	236	40	89	254
237	40	85	254	229
40	81	254	240	40
19	254	225	40	15
254	247	40	69	254
248	40	39	195	162
197	10	0	10	0
205	119	0	254	35
32	54	205	182	24
254	13	202	178	197
254	58	40	184	254
59	40	35	254	44
40	31	205	119	0
24	231	205	119	0
254	34	40	3	35
24	7	205	119	0
254	34	32	249	205
119	0	254	202	40
2	24	191	205	119
0	84	93	254	13
202	176	197	254	58
40	131	254	32	40
236	205	162	24	40
66	254	48	56	13
254	58	48	9	205
119	0	254	32	40
249	24	234	229	62
255	50	140	92	205
142	2	123	254	255
32	248	62	2	205
1	22	237	75	69
92	205	43	45	205
227	45	62	56	215
237	75	71	92	6
0	205	43	45	205
227	45	62	13	215
225	195	162	197	254
13	40	6	254	56
40	2	24	195	43
43	70	43	78	43
43	43	34	93	92
213	205	34	197	24

50	42	83	92	237
91	125	92	126	230
192	32	11	126	184
56	16	192	35	126
185	56	11	201	33
240	216	25	208	17
15	39	201	35	229
42	123	92	25	235
225	197	35	78	35
70	35	9	193	24
213	193	213	42	93
92	229	213	197	209
205	229	25	34	93
92	193	205	43	45
42	93	92	34	91
92	42	81	92	229
62	255	205	1	22
205	227	45	225	205
21	22	42	91	92
34	93	92	42	176
92	70	43	78	235
42	93	92	9	193
167	237	66	235	115
35	114	193	42	93
92	35	35	35	113
35	112	35	35	34
93	92	14	0	126
22	1	205	154	25
254	13	194	61	196
35	126	230	192	202
37	196	42	63	92
58	123	92	237	91
125	92	114	35	115
35	78	35	70	35
9	235	6	0	79
9	235	126	230	192
32	3	121	24	233
225	34	125	92	225
34	123	92	33	0
0	34	69	92	62
1	50	71	92	207
255	42	176	92	70
43	78	9	35	24
185	0	0	0	0

Listado 3

```
9990 REM prepara rutina
9991 INPUT "donde esta la rutina
? ";indice
9992 INPUT "dame el paso entre i
nst.  ";paso
9993 INPUT "primer numero? ";com
ienzo
9994 POKE (indice+286),paso
9995 POKE (indice+288),comienzo-
256*INT (comienzo/256): POKE (in
dice+289),INT (comienzo/256)
9996 RANDOMIZE USR indice
```

CLUB DE USUARIOS MONSER

Si ya tienes tu ordenador (SPECTRUM, MSX, C-64) y deseas estar periódicamente informado de los nuevos programas y periféricos, así como recibir todas las ofertas de software y hardware que, con frecuencia hacemos especialmente para nuestros socios, inscríbete hoy mismo sin ningún gasto, ni compromiso por tu parte.

Enviar el cupón a:

CLUB DE USUARIOS MONSER
Calle Argos, 9. 28037-MADRID

Podrás obtener más información a través de las revistas «48K», «Type and Run», «MSX Soft Magazine» y «Commodore Soft Magazine», o llamando al teléfono (91) 742 72 12/96.

Deseo inscribirme en el CLUB DE USUARIOS MONSER sin ningún gasto, ni compromiso, para recibir información y tener acceso a los descuentos y ofertas especiales del Club.

Nombre ....

Dirección.. Teléfono ....

Ciudad.................................. C.P. .... Provincia ....

Edad.................... Profesión ....

Marca del ordenador.................................. Fecha compra ....

FIRMA

FECHA

Envía inmediatamente el cupón y recibirás en breve un keypanel de regalo para personalizar tus programas.

QUEREMOS VER TUS PROGRAMAS BASIC: SPECTRUM, C64, MSX Y AMSTRAD

Sabemos que eres lo suficientemente ingenioso para no necesitar copiarlos de nadie. Mándanos una cinta de cassette con tu programa, y a ser posible, un listado del mismo.

Premiamos con 5.000 ptas. y un ejemplar de la revista en cuestión, cada programa que publiquemos en cualquiera de nuestras cinco publicaciones.

Envíanos el tuyo, cuanto antes lo hagas más posibilidades tendrás de verlo publicado.

MONSER, S.A.
C/ Argos, 9
28037-MADRID

Deprisa, deprisa...